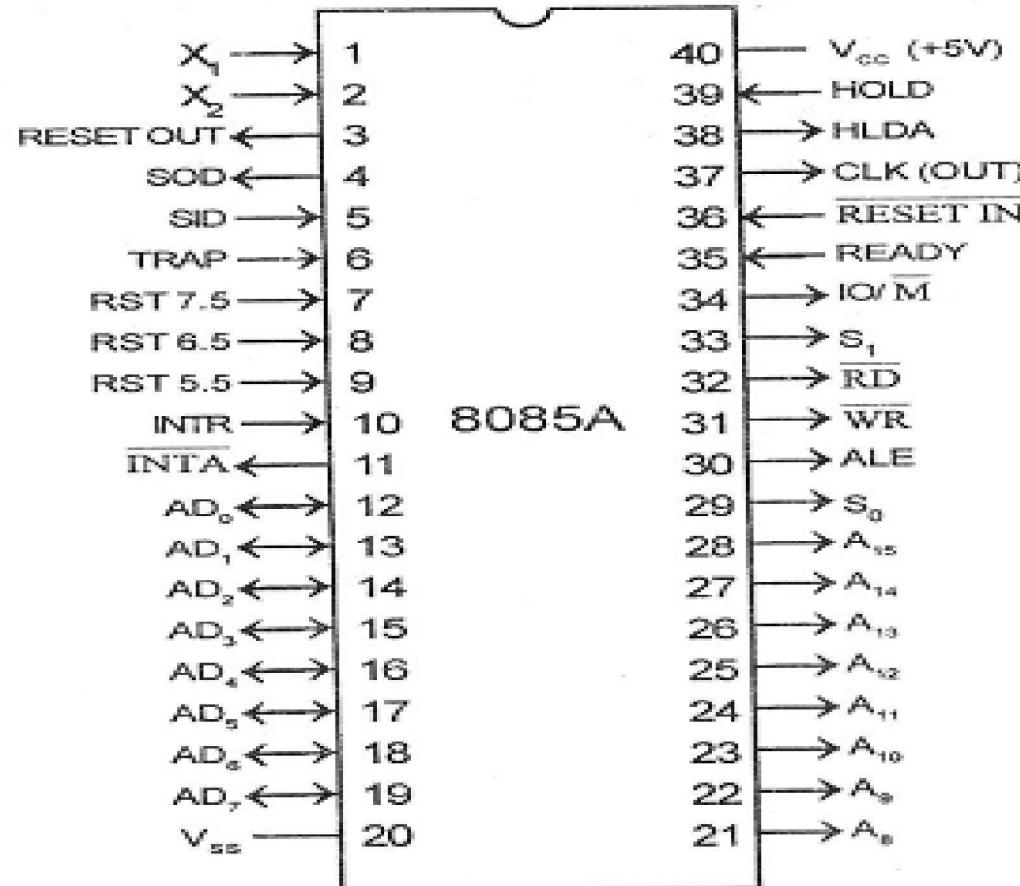


# Timing and state diagram and Memory interfacing of 8085 Microprocessor



# The Address and Data Busses

- The address bus has 8 signal lines **A8 – A15** which are **unidirectional**.
- The other 8 address bits are **multiplexed** (time shared) **with the 8 data bits**.
  - So, the bits **AD0 – AD7** are **bi-directional** and serve as **A0 – A7** and **D0 – D7** at the same time.
    - During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.
    - In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

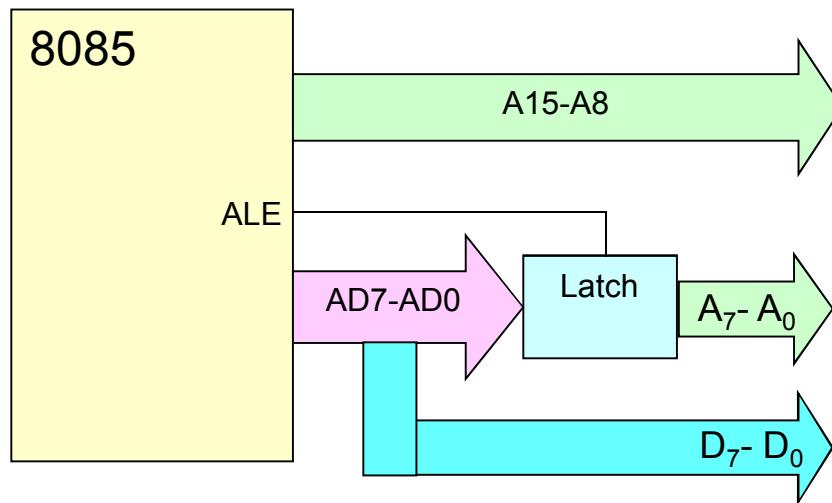
# The Control and Status Signals

There are **4** main **control** and **status** signals:

<b>IO/M</b>	<b>S<sub>1</sub></b>	<b>S<sub>0</sub></b>	<b>Operation performed by the B0RS</b>
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	I/O WRITE
1	1	0	I/O READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

- **ALE: Address Latch Enable.** This signal is a pulse that become **1** when the **AD0 – AD7** lines have an **address** on them. It becomes 0 after that. This signal can be used to enable a latch to save the address bits from the AD lines. →
- **RD: Read.** Active low. **WR: Write.** Active low.
- **IO/M:** This signal specifies whether the operation is a **memory operation** (**IO/M=0**) or an **I/O operation** (**IO/M=1**).
- **S<sub>1</sub> and S<sub>0</sub>** : Status signals to specify the **kind of operation** being performed

# Demultiplexing AD7-AD0



- ALE operates as a pulse during T1, we will be able to latch the address. Then when ALE goes low, the address is saved and the AD7– AD0 lines can be used for their purpose as the bi-directional data lines.

# Cycles and States

- **T- State:** One subdivision of an operation. A T-state lasts for one clock period. An instruction's execution length is usually measured in a number of T-states. (clock cycles).
  - **Machine Cycle:** The time required to complete one operation of accessing memory, I/O, or acknowledging an external request.
    - This cycle may consist of 3 to 6 T-states.
  - **Instruction Cycle:** The time required to complete the execution of an instruction.
    - In the 8085, an instruction cycle may consist of 1 to 6 machine cycles.
- 
- The  $\mu$ P operates with reference to clock signal. The rise and fall of the pulse of the clock gives one clock cycle.
  - Each clock cycle is called a T state and a collection of several T states gives a machine cycle.
  - Important machine cycles are :
    1. Op-code fetch.
    2. Memory read.
    3. Memory write.
    4. I/O-read.
    5. I/O write.

# TIMING AND STATE DIAGRAM

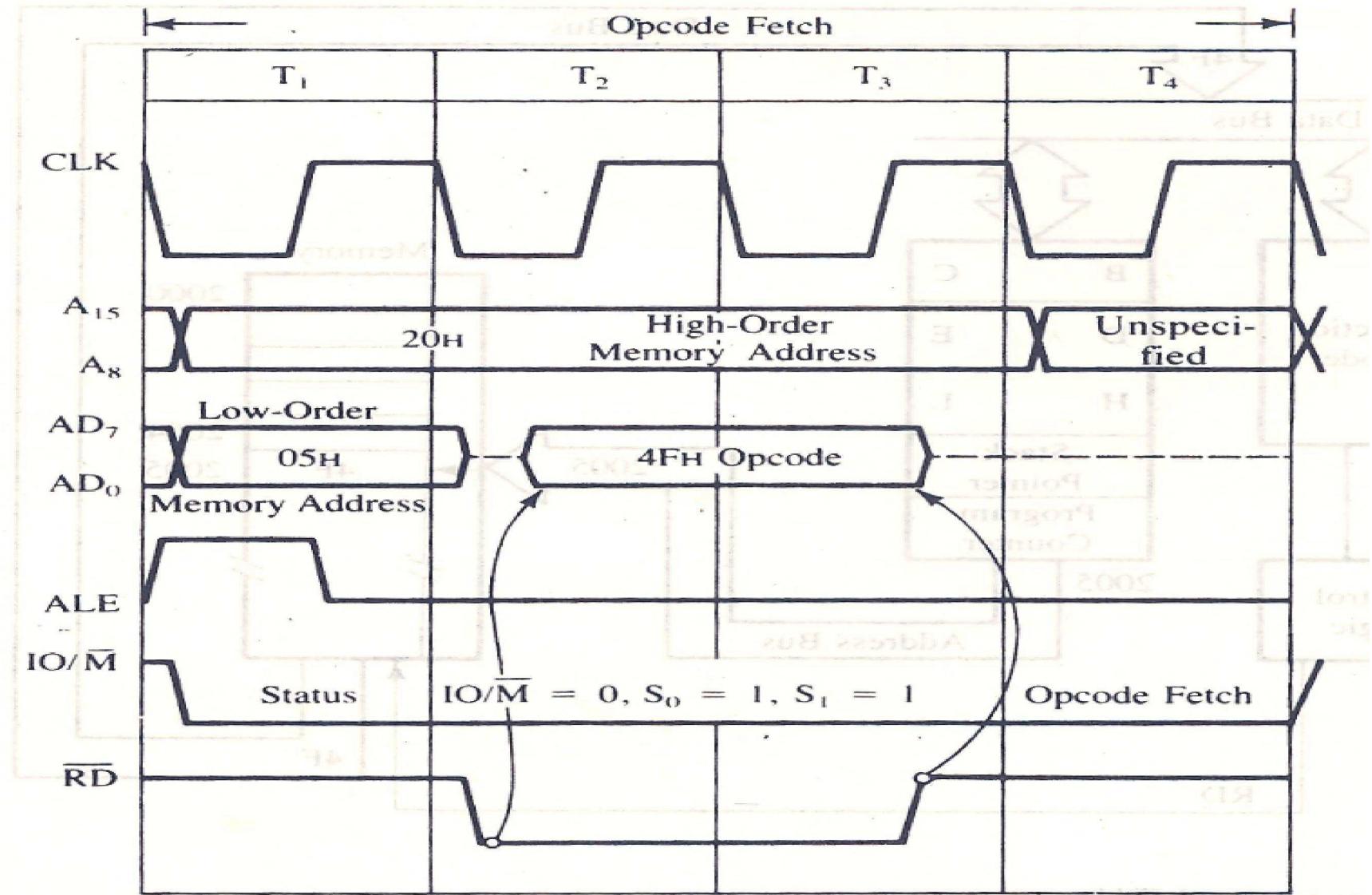
## Op-code Fetch:

- It basically requires 4 T states from  $T_1-T_4$
- The ALE pin goes high at first T state always.
- $AD_0-AD_7$  are used to fetch OP-code and store the lower byte of Program Counter.
- $A_8-A_{15}$  store the higher byte of the Program Counter while  $IO/M^-$  will be low since it is memory related operation.
- $RD^-$  will only be low at the Op-code fetching time.
- $WR^-$  will be at HIGH level since no write operation is done.
- $S_0=1, S_1=1$  for Op-code fetch cycle.

$IO/M^-$	$S_1$	$S_0$	Operation performed by the 8085
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

# TIMING AND STATE DIAGRAM

Op-code fetch cycle :



# TIMING AND STATE DIAGRAM

## Memory Read Cycle:

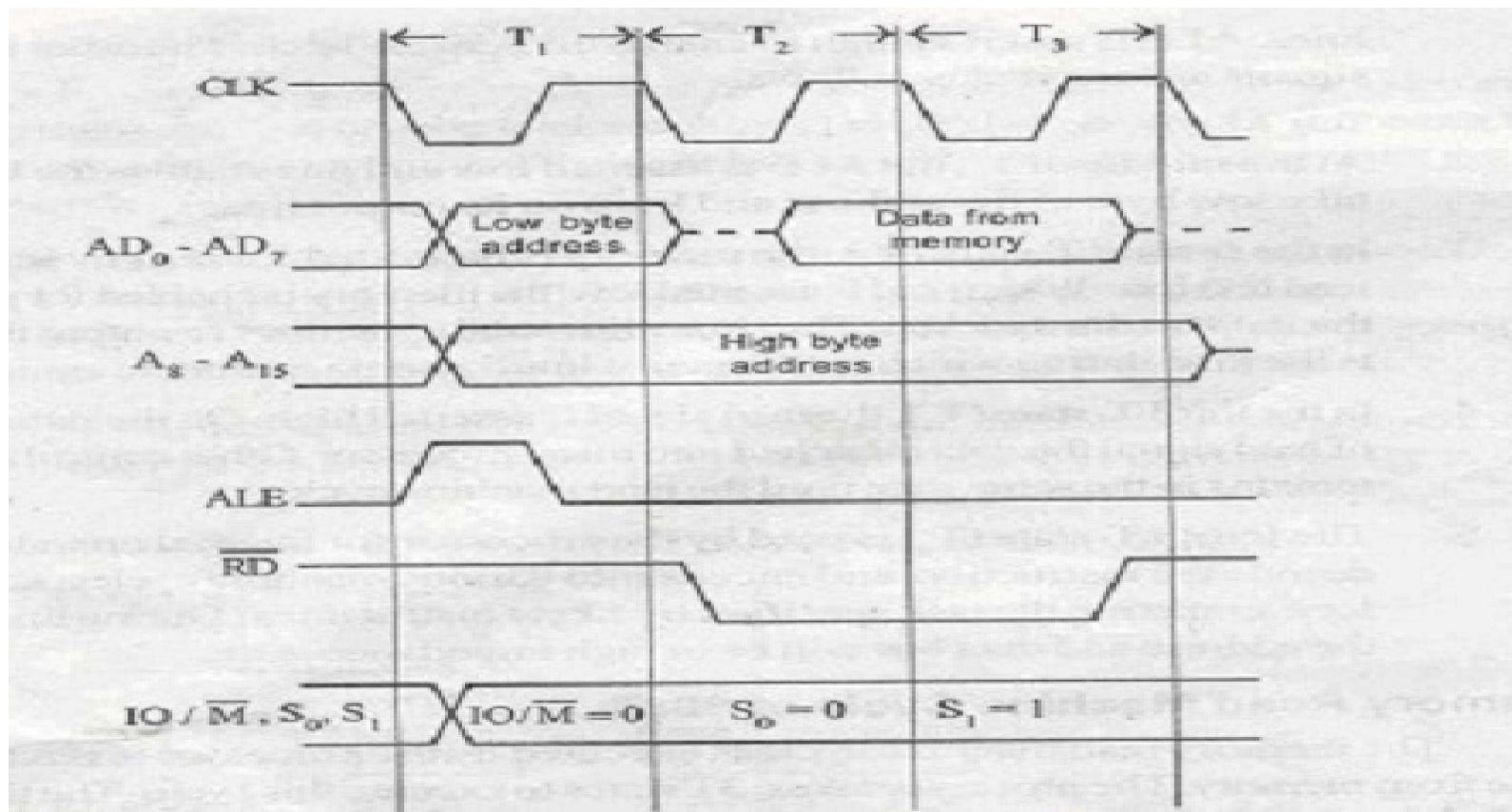
It basically requires 3T states from T<sub>1</sub>-T<sub>3</sub>.

- The ALE pin goes high at first T state always.
- AD<sub>0</sub>-AD<sub>7</sub> are used to fetch data from memory and store the lower byte of address.
- A<sub>8</sub>-A<sub>15</sub> store the higher byte of the address while IO/M<sup>-</sup> will be low since it is memory related operation.
- RD<sup>-</sup> will only be low at the data fetching time.
- WR<sup>-</sup> will be at HIGH level since no write operation is done.
- S<sub>0</sub>=0,S<sub>1</sub>=1 for Memory read cycle.

IO/M	S <sub>1</sub>	S <sub>0</sub>	Operation performed by the 8085
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

# TIMING AND STATE DIAGRAM

Memory Read Cycle:



# TIMING AND STATE DIAGRAM

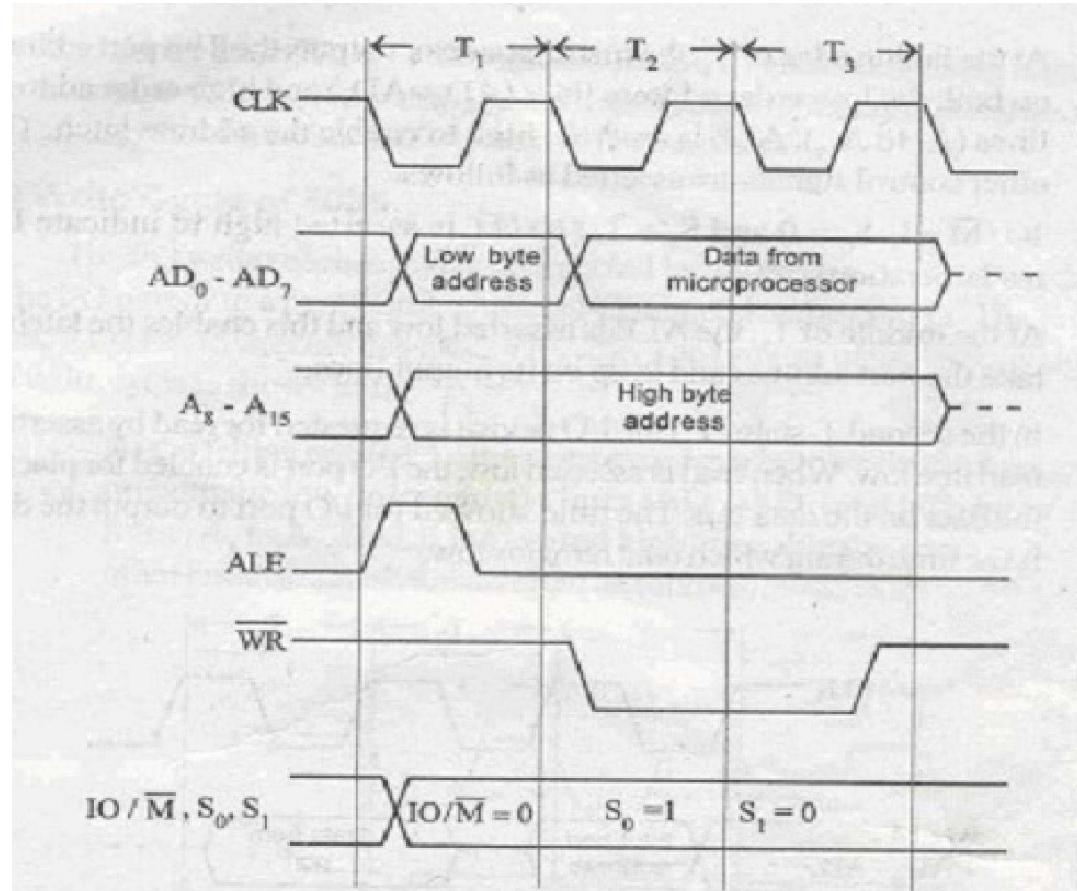
## Memory write Cycle:

- It basically requires 3T states from  $T_1-T_3$ .
- The ALE pin goes high at first T state always.
- $AD_0-AD_7$  are used to fetch data from CPU and store the lower byte of address.
- $A_8-A_{15}$  store the higher byte of the address while  $IO/M^-$  will be low since it is memory related operation.
- $RD^-$  will be HIGH since no read operation is done.
- $WR^-$  will be at LOW level only when data fetching is done.
- $S_0=1, S_1=0$  for Memory write cycle.

$IO/M$	$S_1$	$S_0$	Operation performed by the 8085
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

# TIMING AND STATE DIAGRAM

Memory write Cycle:



# TIMING AND STATE DIAGRAM

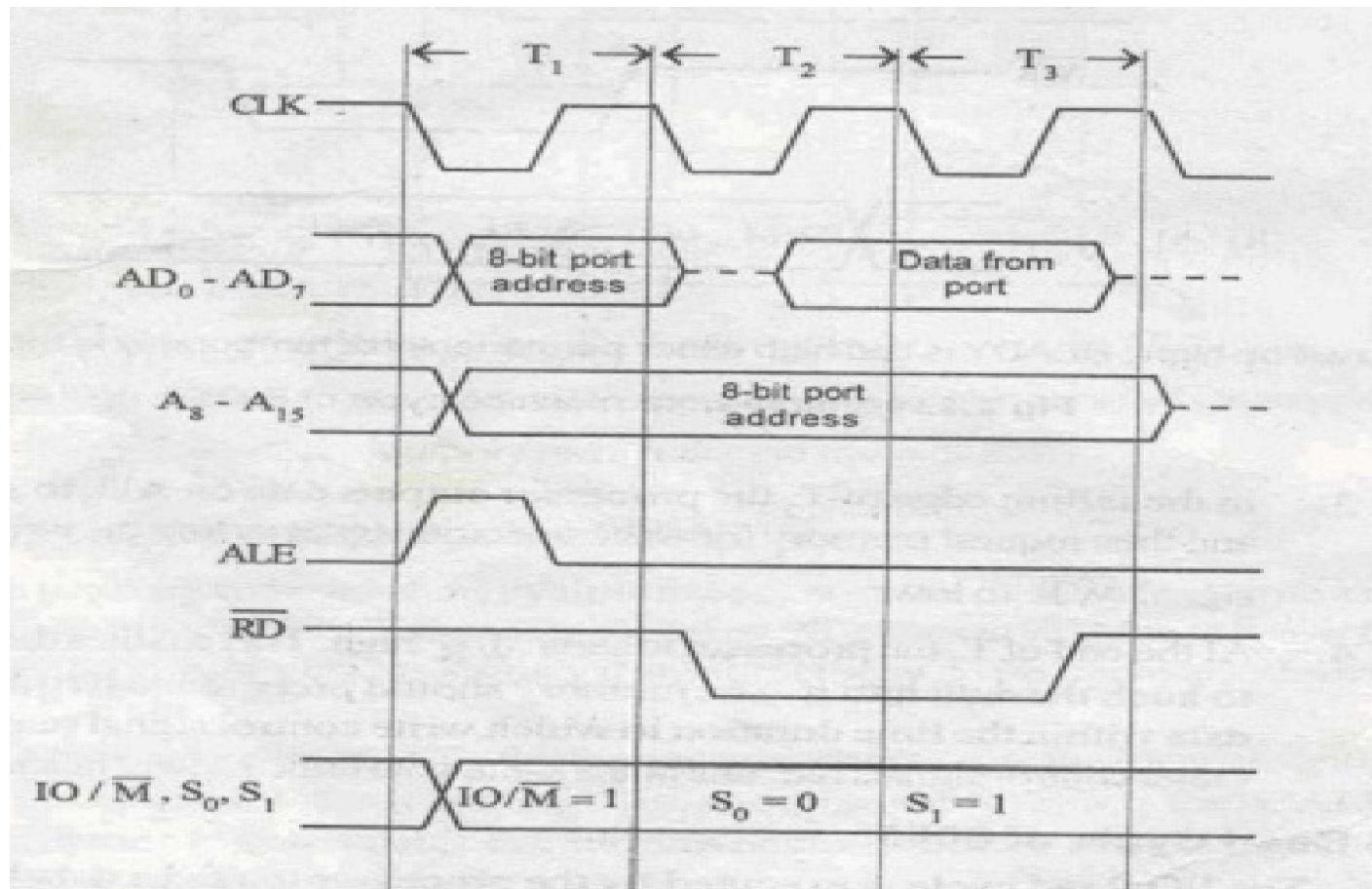
## I/O read Cycle:

It basically requires 3T states from T<sub>1</sub>-T<sub>3</sub>.

- The ALE pin goes high at first T state always.
- At falling edge of T1 the microprocessor outputs the bit address on both lower order lines AD<sub>0</sub>-AD<sub>7</sub> and higher order address A<sub>8</sub>-A<sub>15</sub>
- while IO/M<sup>-</sup> will be high since it is I/O related operation.
- In the second T state RD<sup>-</sup> will be low ,the I/O port is enabled for placing data on the data bus.
- WR<sup>-</sup> will be at HIGH level since no write operation is done.
- S<sub>0</sub>=0,S<sub>1</sub>=1 for I/O read cycle.

# TIMING AND STATE DIAGRAM

I/O read Cycle:



# TIMING AND STATE DIAGRAM

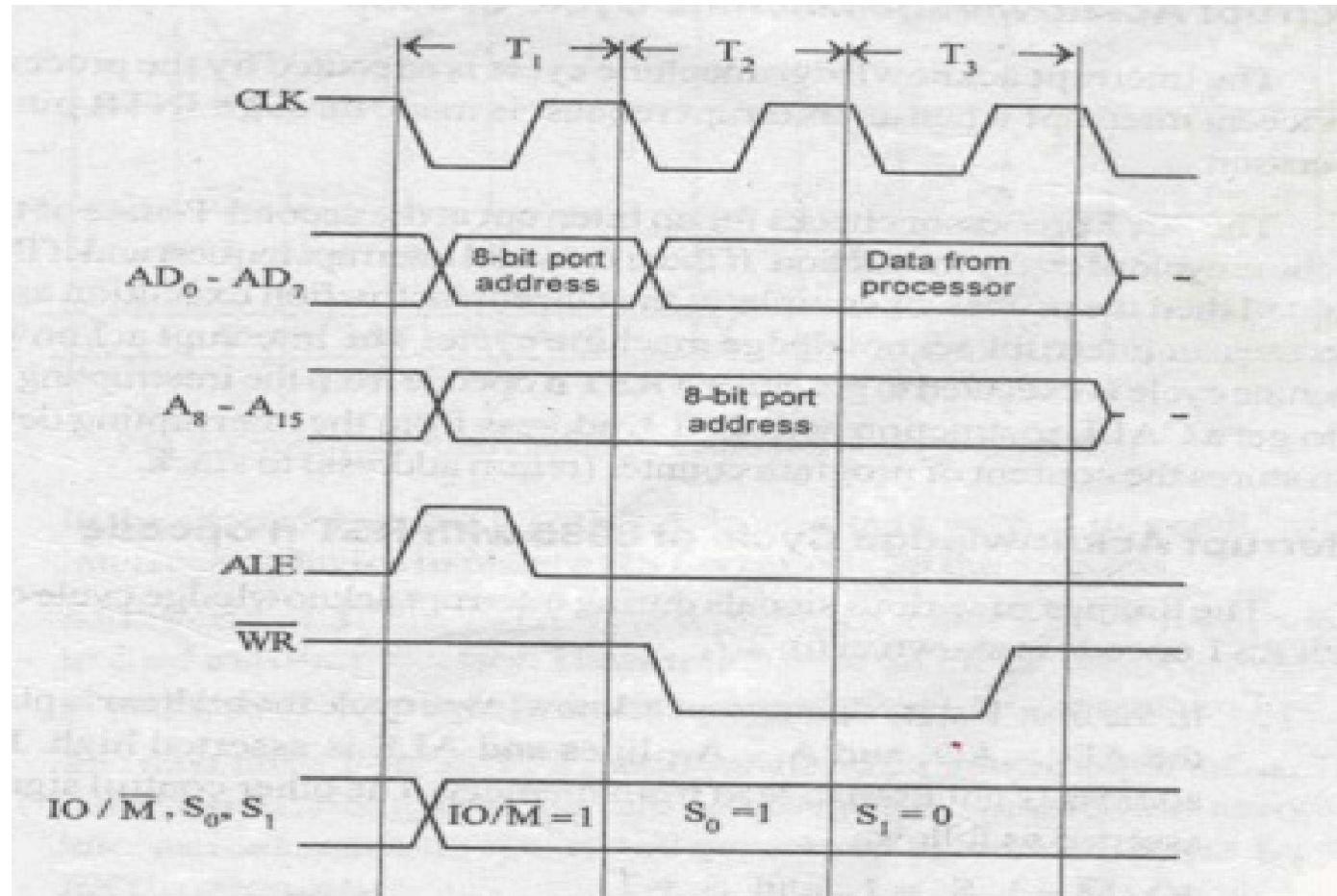
## I/O write Cycle:

It basically requires 3T states from T<sub>1</sub>-T<sub>3</sub>.

- The ALE pin goes high at first T state always.
- At falling edge of T1 the microprocessor outputs the bit address on both lower order lines AD<sub>0</sub>-AD<sub>7</sub> and higher order address A<sub>8</sub>-A<sub>15</sub>
- while IO/M<sup>-</sup> will be high since it is I/O related operation.
- RD<sup>-</sup> will be HIGH since no read operation is done.
- WR<sup>-</sup> will be at LOW level only when data fetching is done.
- S<sub>0</sub>=1,S<sub>1</sub>=0 for Memory write cycle.

# TIMING AND STATE DIAGRAM

I/O write Cycle:



## summary:

1. Opcode fetch .....	- 4 / 6 T
2. Memory Read .....	- 3 T
3. Memory Write .....	- 3 T
4. I/O Read .....	- 3 T
5. I/O Write .....	- 3 T
6. Interrupt Acknowledge .....	- 6 / 12 T
7. Bus Idle .....	- 2 / 3 T

### Delay routine process:

A delay routine is generally written as a subroutine .In delay routine a count or number is loaded in a register of microprocessor. Then it is decremented by one and the zero flag is checked to verify whether the content of register is zero or not .this process is continued until the content of register is zero. when it is zero the time delay is over and the control is transferred to main program to carry out the desired operation.

The delay time can be calculated by multiplying the total number of T-states required to execute subroutine and the time for one T-state of the processor. The time for T-state of the processor is given by inverse of the internal clock frequency of the processor.  
If microprocessor has 5 MHz clock then internal clock frequency =  $5/2 = 2.5 \text{ MHz}$   
Time for one T-state=  $1/2.5 \text{ MHz} = 0.4 \mu\text{sec}$

Example delay routine Write a delay routine to produce a time delay of 0.5 msec in microprocessor whose clock source is 6MHz quartz crystal.

Delay routine:

	MVI D,N	;load count value N register D
LOOP :	DCR D	;decrement the count
	JNZ LOOP	;if count is not zero form loop
	RET	;if count is zero,then return to main program

<i>Instruction</i>	<i>T-State required for execution of an instruction</i>	<i>Number of times the instruction is executed</i>	<i>Total T-States</i>
CALL addr16	18	1	$18 \times 1 = 18$
MVI D, N	7	1	$7 \times 1 = 7$
DCR D	4	N times	$4 \times N = 4N$
JNZ LOOP	10 (or)	(N-1) times	$10 \times (N-1) = 10N - 10$
	7	1	$7 \times 1 = 7$
RET	10	1	$10 \times 1 = 10$
<b>TOTAL T-STATES FOR DELAY SUBROUTINE</b>			<b><math>14N + 32</math></b>

### Calculation to find the count value, N:

$$\text{External clock frequency} = 6 \text{ Mhz}$$

$$\begin{aligned}\text{Internal clock frequency} &= \text{External Frequency} / 2 \\ &= 6 / 2 \\ &= 3 \text{ Mhz}\end{aligned}$$

$$\begin{aligned}\text{Time period for 1 T-State} &= 1 / \text{Internal clock frequency} \\ &= 1 / 3 \times 10^6 \\ &= 0.333\mu\text{s}\end{aligned}$$

$$\begin{aligned}\text{No. of T-states required for delay of } 0.5\text{mS} &= \text{Required time delay} / \text{Time for one T-state} \\ &= 0.5\text{mS} / 0.333\mu\text{s} \\ &= 1500.10 \\ &\approx 1500 = 1500_{10}\end{aligned}$$

From above table, we know that;

$$14N + 32 = 1500$$

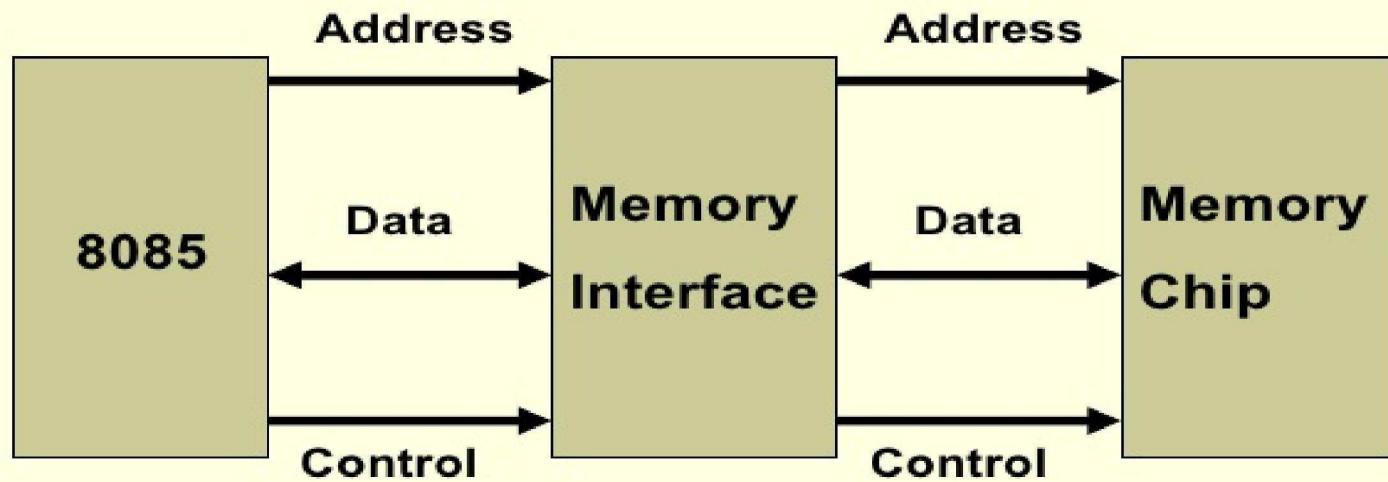
$$N = (1500 - 32) / 14 = 104.857_{10} \approx 105_{10} = 69_H$$

Therefore by replacing the count value, N by  $69_H$  in the above program, a delay of  $0.5\text{mSec}$  can be produced.

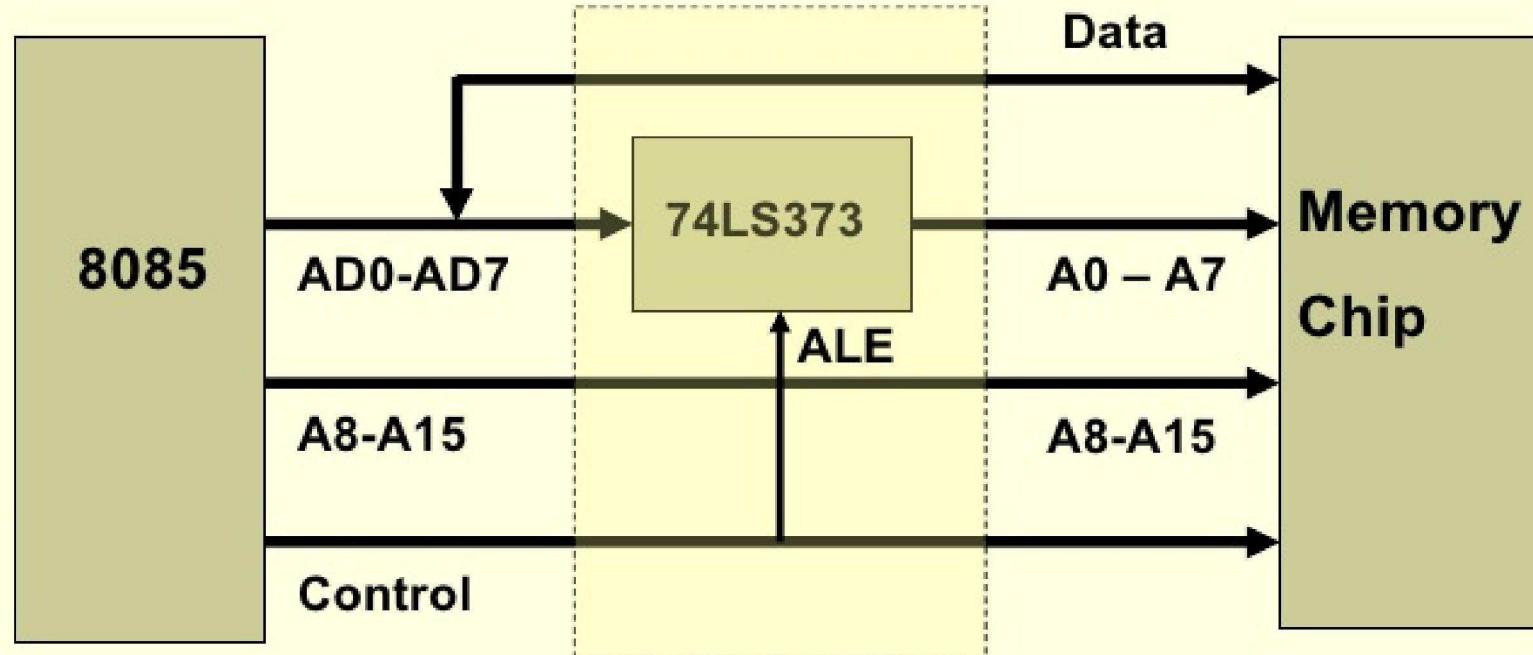
# 8085 Memory Interfacing

- Generally  $\mu$ P 8085 can address 64 kB of memory .
- Generally EPROMS are used as program memory and RAM as data memory.
- We can interface Multiple RAMs and EPROMS to single  $\mu$ P .
- Memory interfacing includes 3 steps :
  1. Select the chip.
  2. Identify register.
  3. Enable appropriate buffer.

## 8085 Interfacing with Memory chips

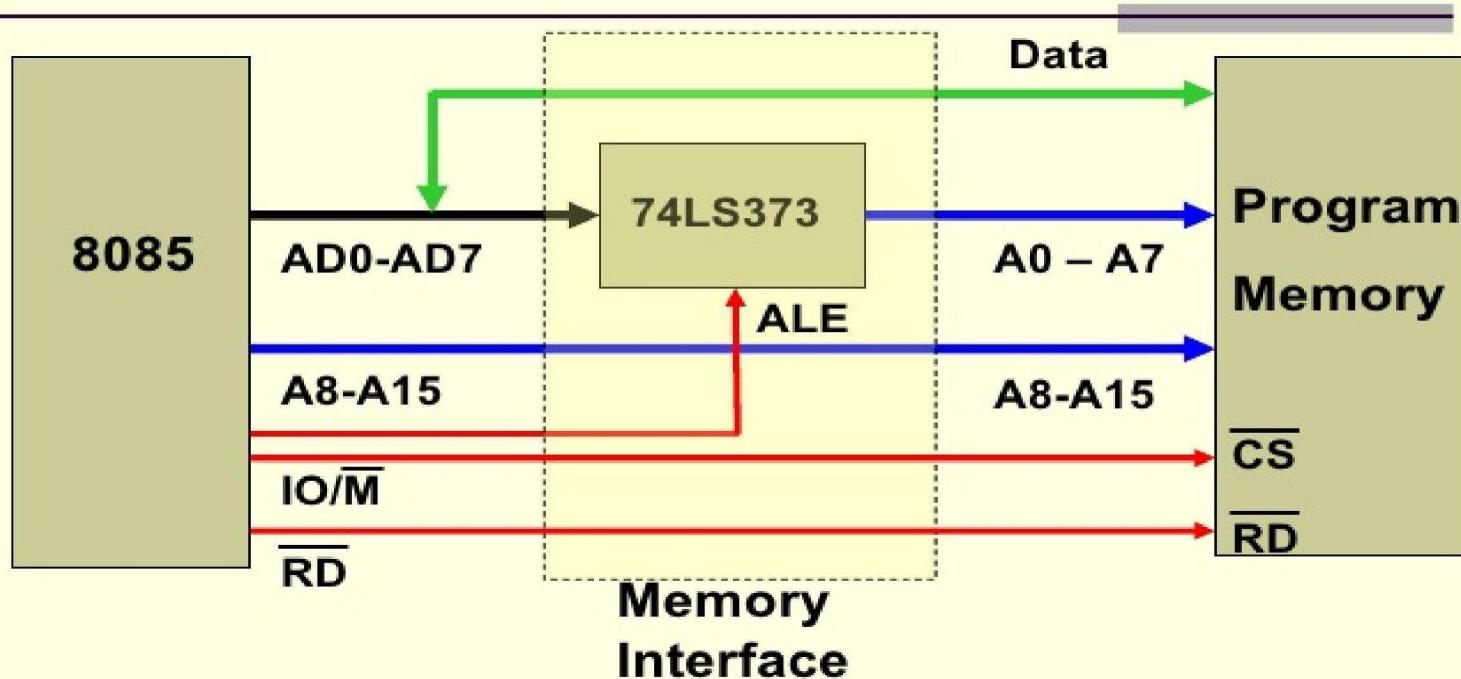


# 8085 Interfacing with Memory chips



**Memory  
Interface**

## 8085 Interfacing with Memory chips



## Memory Mapping

---

- 8085 has 16-bit Address Bus
- The complete address space is thus given by the range of addresses **0000H – FFFFH**
- The range of addresses allocated to a memory device is known as its memory map

## Memory map: 64K memory device

---

- Address lines required: 16 (A0 – A15)
- Memory map: **0000H - FFFFH**

## Memory map: 32K memory device

---

- Address lines required: 15 (A0 – A14)
- Memory map: depends on how address line A15 is connected

- So the memory map is

A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>	A <sub>12</sub>	A <sub>11</sub> to A <sub>0</sub>
0	0	0	0	0.... 0 0

= 0000H

to

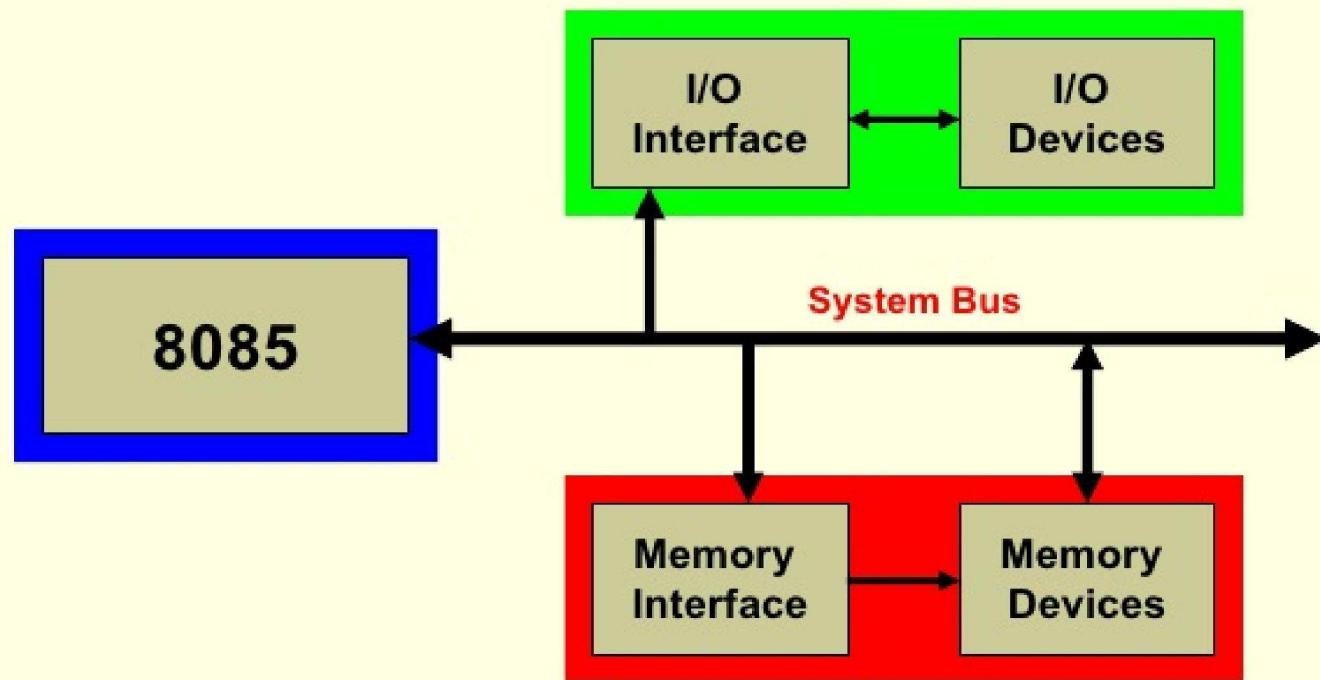
A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>	A <sub>12</sub>	A <sub>11</sub> to A <sub>0</sub>
0	1	1	1	1.... 111

= 7FFFH

# Interfacing I/O devices with 8085

**Peripheral-mapped I/O  
&  
Memory-mapped I/O**

# Interfacing I/O devices with 8085



## Memory-mapped I/O

---

- 8085 uses its **16-bit address bus** to identify a memory location
- Memory address space: **0000H** to **FFFFH**
- 8085 needs to identify I/O devices also
- I/O devices can be interfaced using addresses from memory space
- 8085 treats such an I/O device as a memory location
- This is called Memory-mapped I/O

## Peripheral-mapped I/O

---

- 8085 has a separate **8-bit** addressing scheme for I/O devices
- I/O address space: **00H** to **FFH**
- This is called Peripheral-mapped I/O or I/O-mapped I/O

## 8085 Communication with I/O devices

---

- Involves the following three steps
  1. Identify the I/O device (with address)
  2. Generate Timing & Control signals
  3. Data transfer takes place
- 8085 communicates with a I/O device only if there is a **Program Instruction** to do so

## **1. Identify the I/O device (with address)**

---

1. Memory-mapped I/O (16-bit address)
2. Peripheral-mapped I/O (8-bit address)

## **2. Generate Timing & Control Signals**

---

- Memory-mapped I/O
  - Reading Input:  $\text{IO/M} = 0$ ,  $\overline{\text{RD}} = 0$
  - Write to Output:  $\text{IO/M} = 0$ ,  $\overline{\text{WR}} = 0$
- Peripheral-mapped I/O
  - Reading Input:  $\text{IO/M} = 1$ ,  $\overline{\text{RD}} = 0$
  - Write to Output:  $\text{IO/M} = 1$ ,  $\overline{\text{WR}} = 0$

## **3. Data transfer takes place**

# Peripheral I/O Instructions

---

## ■ **IN Instruction**

- Inputs data from input device into the accumulator
- It is a 2-byte instruction
- Format: **IN 8-bit port address**
- Example: **IN 01H**

---

## ■ **OUT Instruction**

- Outputs the contents of accumulator to an output device
- It is a 2-byte instruction
- Format: **OUT 8-bit port address**
- Example: **OUT 02H**

# Memory-mapped I/O Instructions

---

- I/O devices are identified by 16-bit addresses
- 8085 communicates with an I/O device as if it were one of the memory locations
- Memory related instructions are used
- For e.g. LDA, STA
- **LDA 8000H**
  - Loads A with data read from input device with 16-bit address 8000H
- **STA 8001H**
  - Stores (Outputs) contents of A to output device with 16-bit address 8001H

# 8085 Memory Interfacing

- Example: Interface 2Kbytes of Memory to 8085 with starting address 8000H.

Initially we realize that 2K memory requires 11 address lines ( $2^{11}=2048$ ). So we use  $A_0-A_{10}$ .

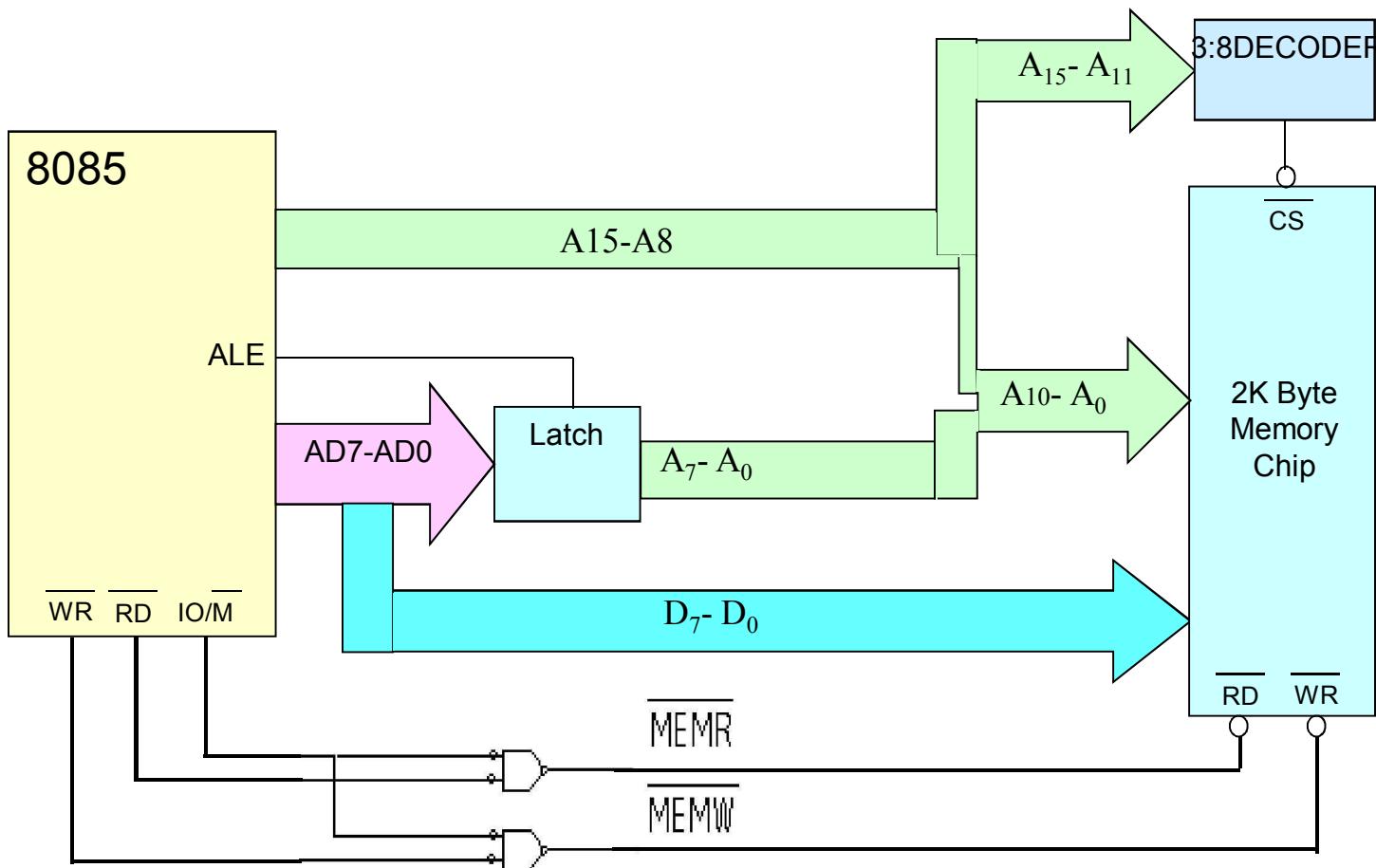
- Write down  $A_{15} - A_0$

# 8085 Memory Interfacing

- Address lines A<sub>0</sub>-A<sub>10</sub> are used to interface memory while A<sub>11</sub>,A<sub>12</sub>,A<sub>13</sub>,A<sub>14</sub>,A<sub>15</sub> are given to 3:8 Decoder to provide an output signal used to select the memory chip CS<sup>-</sup> or Chip select input.
- MEMR<sup>-</sup> and MEMW<sup>-</sup> are given to RD<sup>-</sup> and WR<sup>-</sup> pins of Memory chip.
- Data lines D<sub>0</sub>-D<sub>7</sub> are given to D<sub>0</sub>-D<sub>7</sub> pins of the memory chip.
- In this way memory interfacing can be achieved.

# 8085 Memory Interfacing

- The diagram of 2k interfacing is shown below:



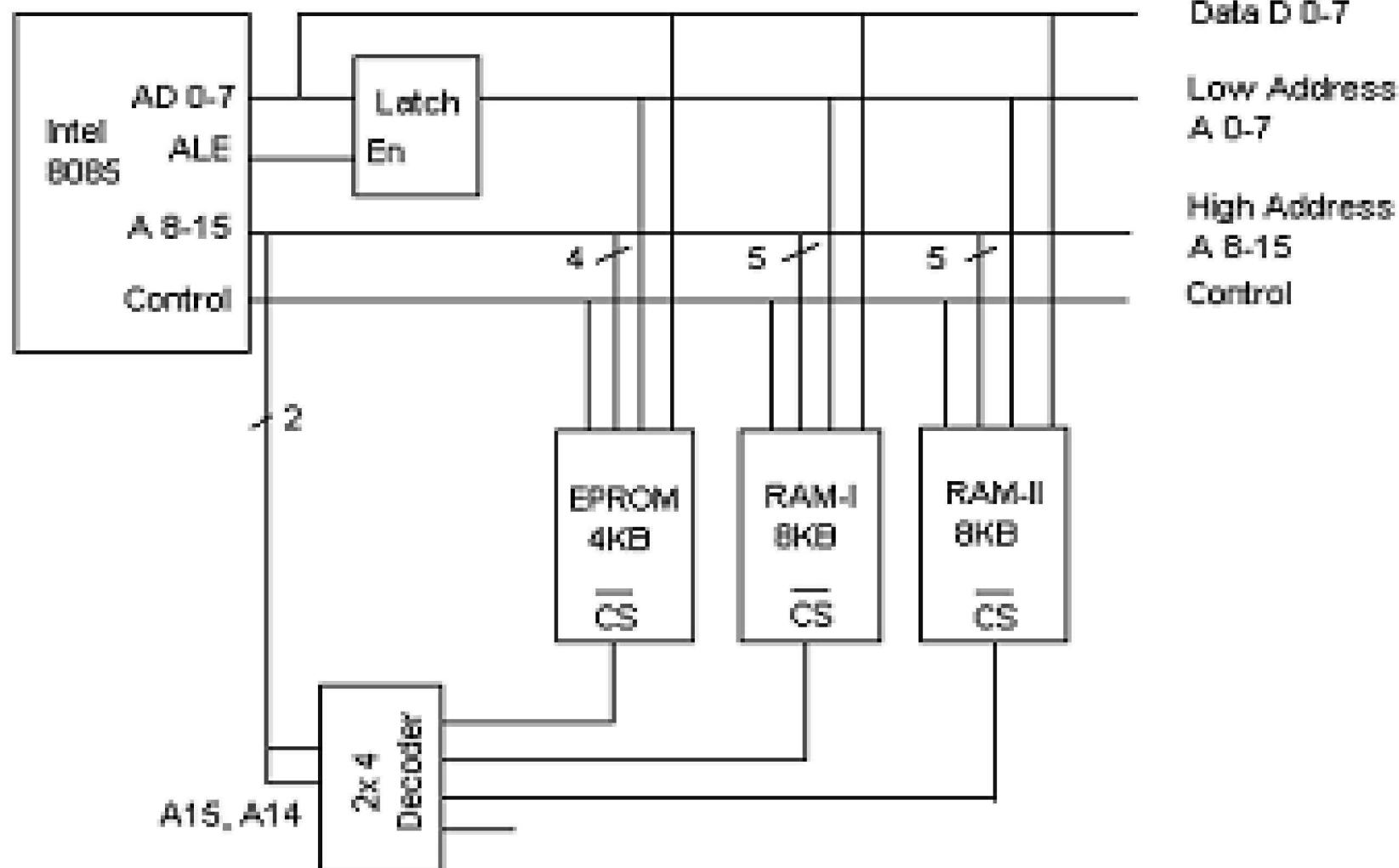
**Draw the circuit diagram of an 8085 system, having a 4 KB EPROM and two 8 KB RAM ICs. The starting address of the EPROM is 0000H and that of RAM is 8000H. The address of the decoder circuits should be clearly shown.**

**Answer 3**

- |               |   |  |
|---------------|---|--|
| <b>EPROM</b>  | - | 4 KB (Address lines required is 12 – A <sub>0</sub> to A <sub>11</sub> ) |
| <b>RAM-I</b>  | - | 8 KB (Address lines required is 13 – A <sub>0</sub> to A <sub>12</sub> ) |
| <b>RAM-II</b> | - | 8 KB (Address lines required is 13 – A <sub>0</sub> to A <sub>12</sub> ) |

## Mapping of Addresses to Memory Ics

# 8085 Memory Interfacing



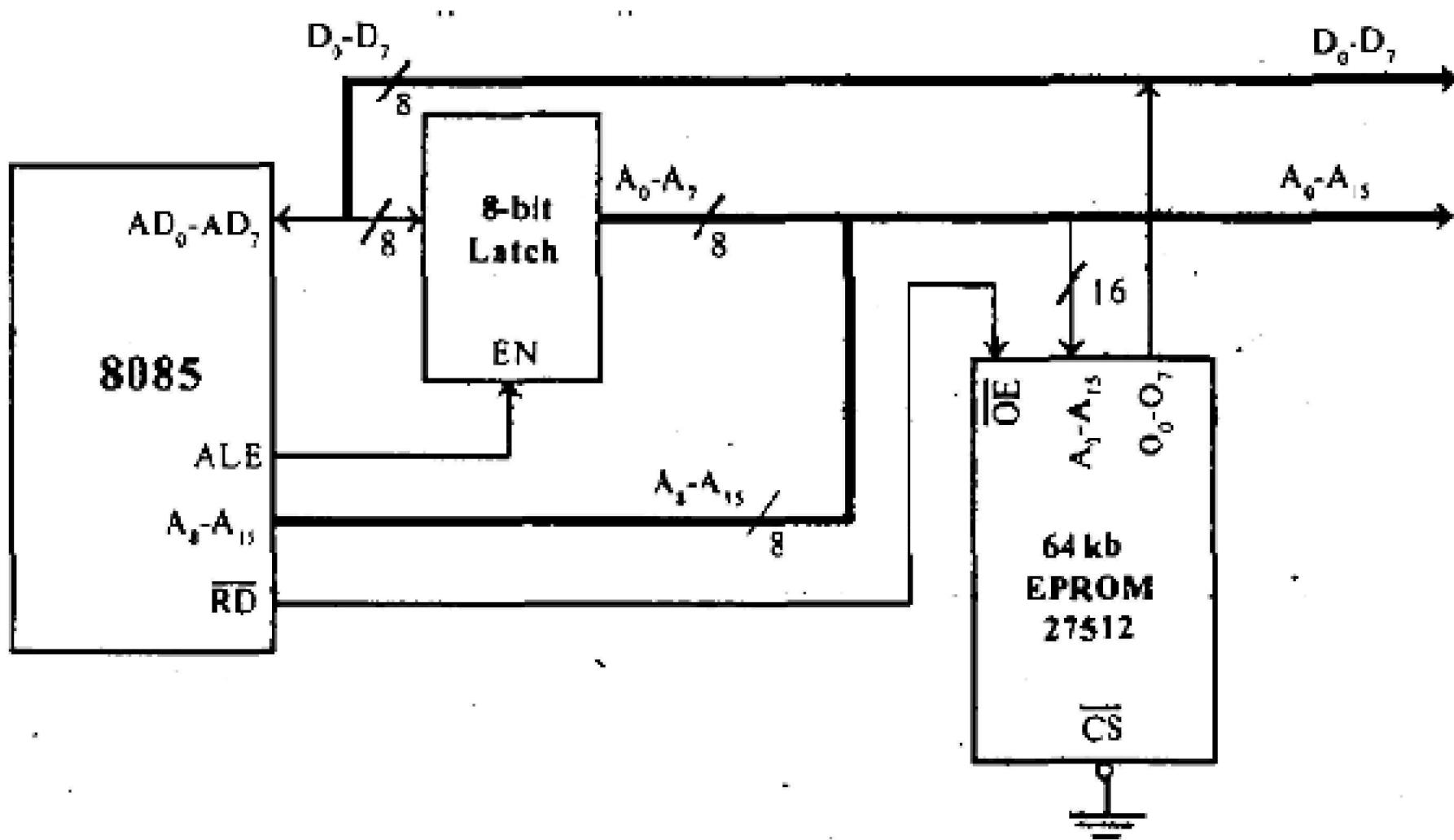
## **EXAMPLE**

**Consider a system in which the full memory space 64KB is utilized for EPROM memory. Interface the EPROM with 8085 processor.**

**The memory capacity is 64 Kbytes. i . e  $2^n = 64 \times 1000$  bytes where n = address lines . So, n = 16.**

**In this system the entire 16 address lines of the processor are connected to address input pins of memory IC in order to address the internal locations of memory.**

**The chip select (CS) pin of EPROM is permanently tied to logic low (i.e., tied to ground). Since the processor is connected to EPROM, the active low RD pin is connected to active low output enable pin of EPROM. The range of address for EPROM is 0000H to FFFFH.**



**Example :**

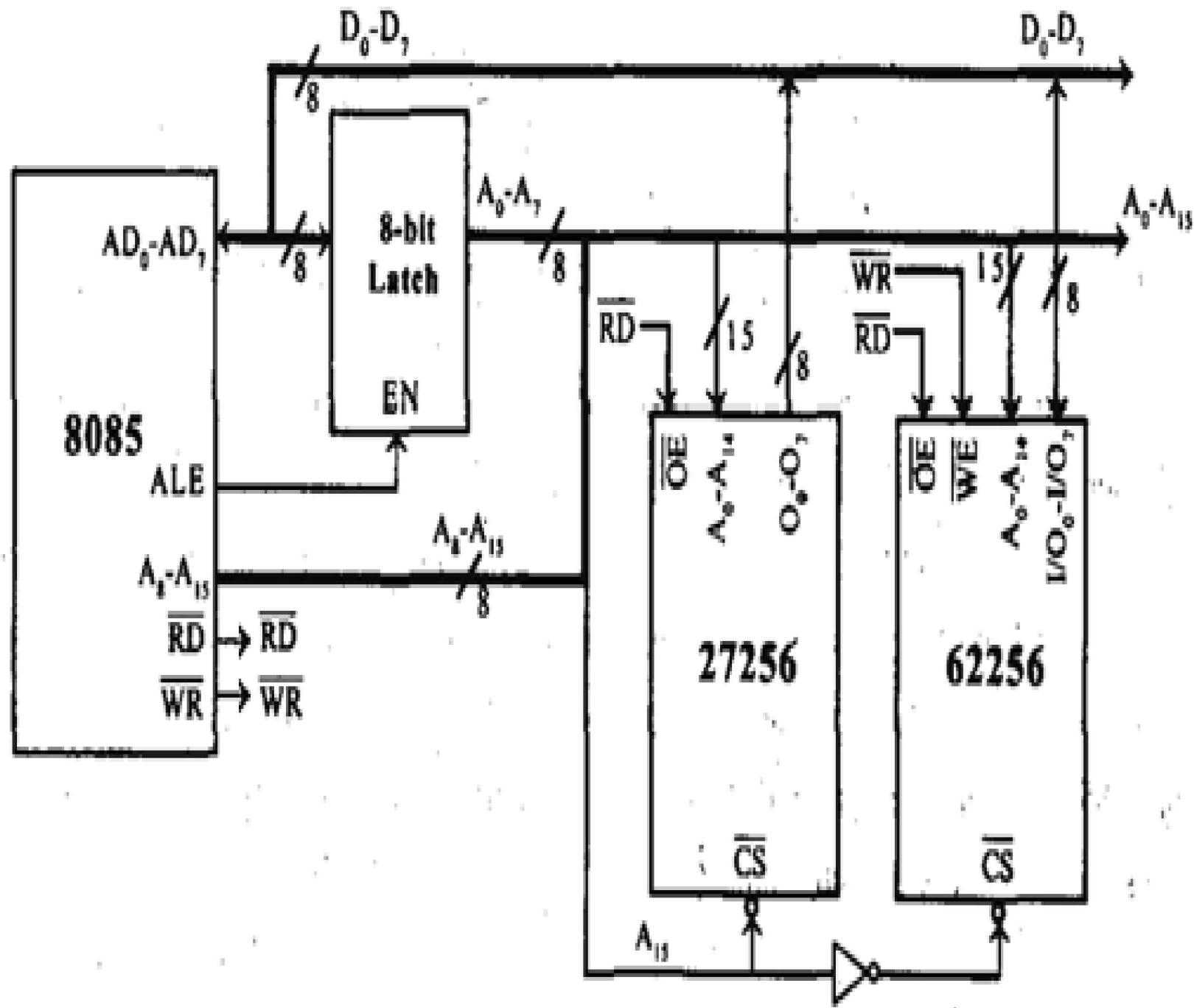
Consider a system in which the available 64KB memory space is equally divided between EPROM and RAM. Interface the EPROM and RAM with 8085 processor. Implement 32kb memory capacity of EPROM using single IC 27256. 32kb RAM capacity is implemented using single IC 62256.

The 32kb memory requires 15 address lines and so the address lines A0 - A14 of the processor are connected to 15 address pins of both EPROM and RAM.

The unused address line A15 is used as to chip select. If A15 is 1, it select RAM and If A15 is 0, it select EPROM. Inverter is used for selecting the memory.

The memory used is both RAM and EPROM, so the low RD and WR pins of processor are connected to low WE and OE pins of memory respectively.

The address range of EPROM will be 0000H to 7FFFH and that of RAM will be 7FFFH to FFFFH.



**Example:**

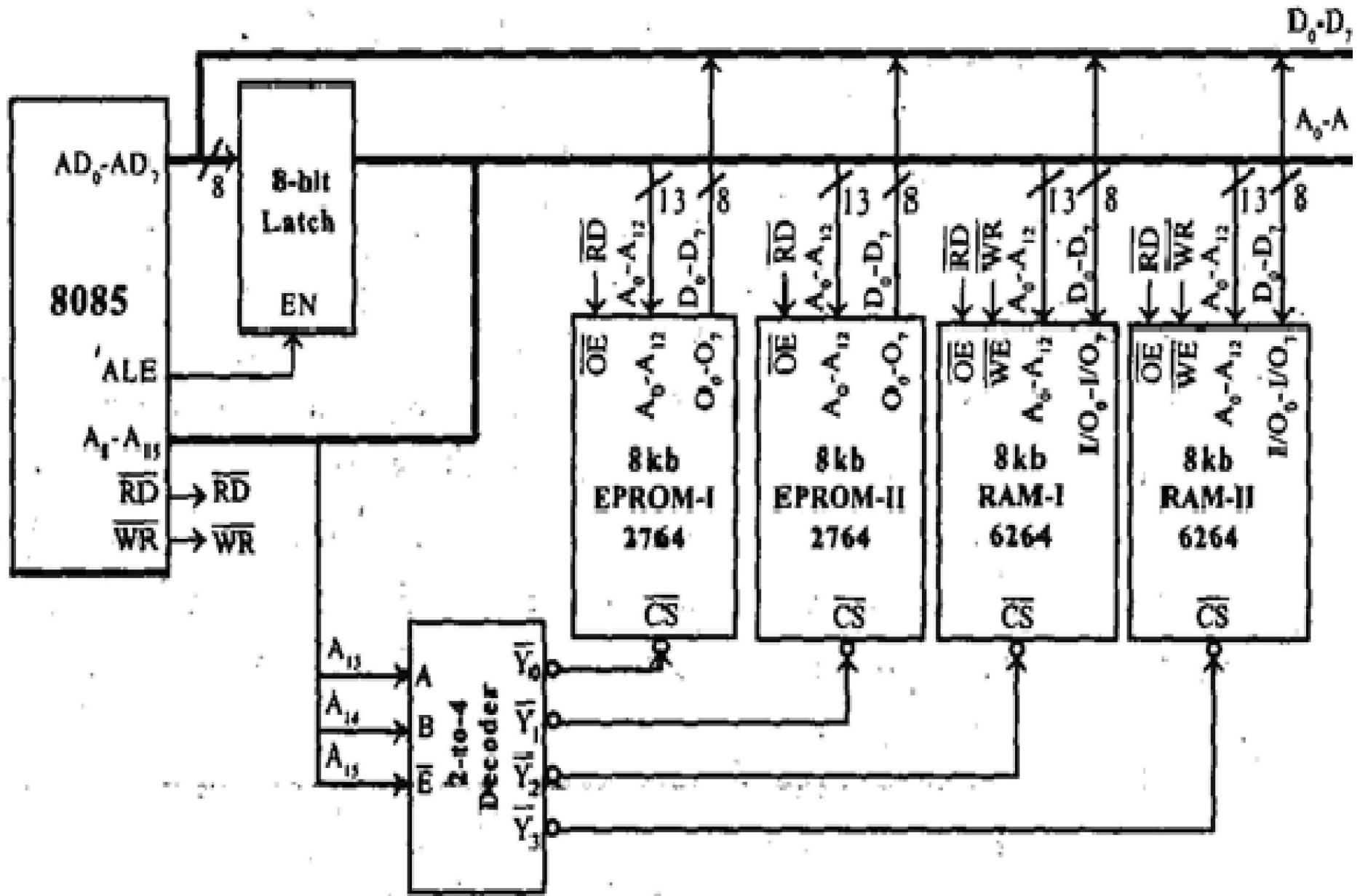
Consider a system in which 32kb memory space is implemented using four numbers of 8KB memory. Interface the EPROM and RAM with 8085 processor.

The total memory capacity is 32KB. So, let two number of 8KB in memory be EPROM and the remaining two numbers be RAM. Each 8KB memory requires 13 address lines and so the address lines A0- A12 of the processor are connected to 13 address pins of all the memory. The address lines A13 - A14 can be decoded using a 2-to-4 decoder to generate four chip select signals.

These four chip select signals can be used to select one of the four memory IC at any one time.

The address line A15 is used as enable for decoder.

The simplified schematic memory organization is shown.



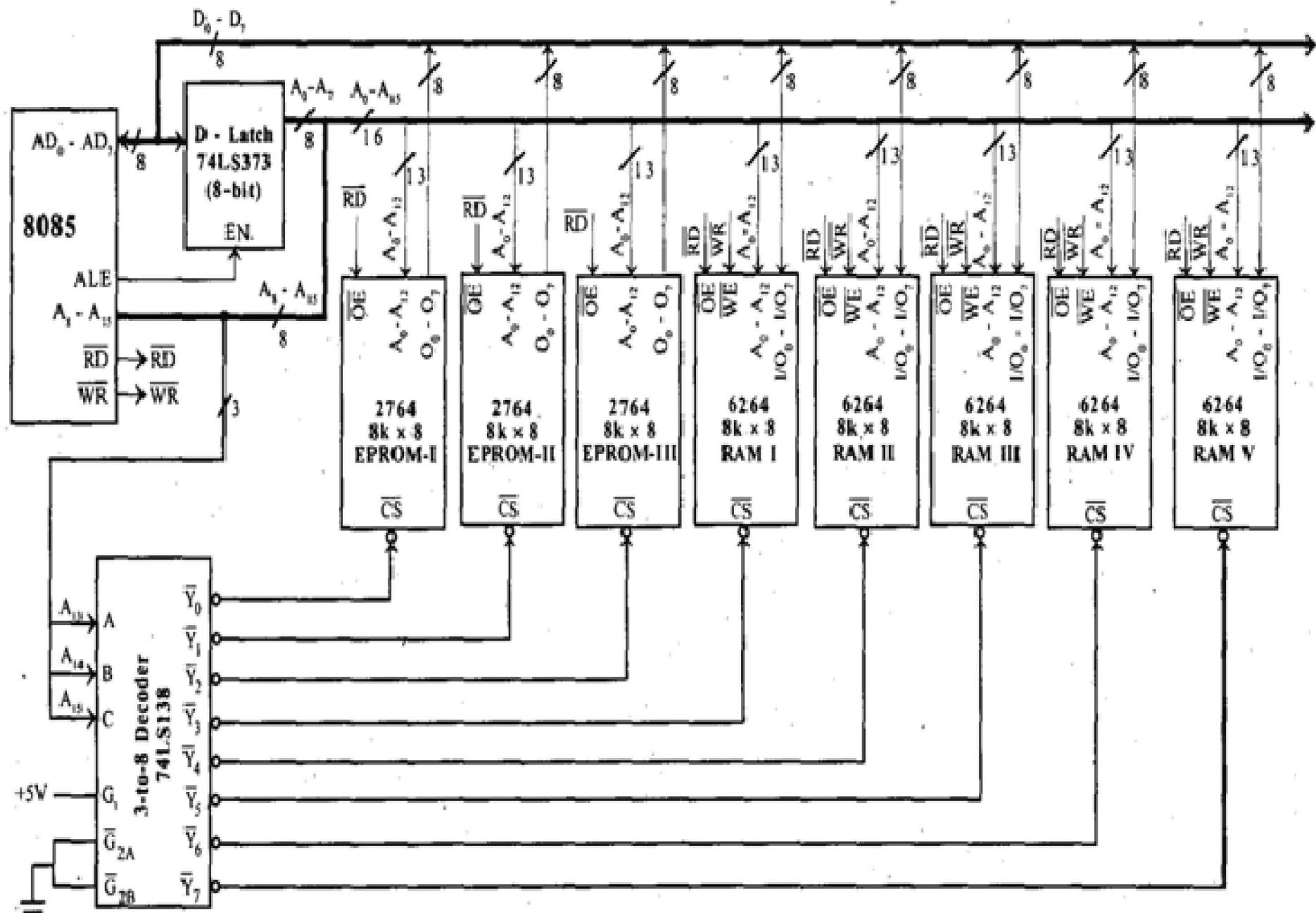


**Example :**

Consider a system in which the 64KB memory space is implemented using eight numbers of 8KB memory. Interface the EPROM and RAM with 8085 processor. The total memory capacity is 64KB. So, let 3 numbers of 8KB EPROM and 5 numbers of 8KB RAM.

Each 8KB memory requires 13 address lines. So the address line A0 - A12 of the processor are connected to 13 address pins of all the memory ICs. The address lines A13, A14 and A15 are decoded using a 3-to-8 coder to generate eight chip select signals. These eight chip select signals can be used to select one of the eight memories at any one time.

The memory interfacing is shown in following figure.





## Distinguish I/O mapped I/O and memory mapped I/O.

Mapping is the process by which the addresses are allocated to the I/O devices.

The two kinds of mapping are

- a) Memory mapped I/O
- b) I/O mapped I/O

<u>S.No</u>	<i>Memory mapped I/O</i>	<i>I/O mapped I/O</i>
1	<i>16 bit address is given to each I/O device</i>	<i>8 bit address is given to each I/O device</i>
2	<i>Each I/O device is treated like a memory location and they are accessed using instructions related to memory operations.</i>	<i>All I/O devices are accessed using only two instructions viz., IN and OUT.</i>
3	<i>Data can be transferred between I/O devices and all registers in <math>\mu</math> P.</i>	<i>Data can be transferred only between I/O devices and accumulator in <math>\mu</math> P.</i>
4	<i>This scheme is used in system, where memory requirement is small.</i>	<i>This scheme is used in system, where complete memory capacity is required.</i>
5	<i>Only Memory Read &amp; Write machine cycles are involved during data transfer with I/O devices.</i>	<i>Only I/O Read &amp; Write machine cycles are involved during data transfer with I/O devices.</i>
6	<i>Large number of I/O devices can be connected in this scheme.</i>	<i>Only maximum of <math>256 (=2^8)</math> I/O devices can be connected in this scheme.</i>